Olympiads in Informatics, 2014, Vol. 8, 21–34

Programming Trainings and Informatics Teaching Through Online Contests

Sébastien COMBÉFIS, Jérémy WAUTELET
Computer Science and IT in Education ASBL, Belgium
e-mail: sebastien@combefis.be, jeremy.wautelet@csited.be

Abstract. Promoting computer science though programming is widespread all around the world. However, there are not always enough human resources to support trainings and teaching of programming. At the same time, online programming contests have also spread and are getting accessible to people at large. This paper is about how it is possible to use online programming contests to build trainings and to support the teaching of programming. The paper first reviews how programming contests can be classified. It then proposes classification criteria and applies them to a selection of existing online programming contests. Based on that classification criteria and review, the paper discusses how such contests can be used to build programming trainings and also to support teaching. Finally, the paper presents an online platform that allows people to create a contestant profile to compare them to other users of the platform and to discuss about the contests they took part in. All this work aims at increasing the motivation of people when learning to program and at promoting computer science among young people, with limited human resources and using online social connections between people.

Keywords: programming contest, learning programming, contestant's profile.

1. Introduction

There are many different kinds of online programming contests. The main goal of such contests is to allow contestants or teams of contestants to compete. Contests can be of many different kinds: finding the most efficient algorithm, modelling a challenging problem and writing a program to solve it, developing an artificial intelligence (AI) algorithm to be run against AIs from other contestants…

In addition to the contest itself and its main goal, participating to such contests is also a way for contestants to learn and improve their own skills. The better and more relevant the support to the contestant and the received feedback are, the more his/her learning will be of good quality. For example, providing contestants with the correct solution annotated with explanations at the end of the contest, or allowing them to participate in teams may improve their learning.

Of course, given limited human resources to train, teach and coach contestants, it is not always possible to provide individual feedbacks to all the contestants, or to provide

precise and detailed enough feedback for all the tasks that the contestants had to solve for a given contest.

This paper proposes to use existing online contests to support teaching programming skills, and more generally to promote informatics. The focus is not only on contestants that would like to improve their skills, but also to anyone who wants to start learning programming. Depending on the precise purpose and on what is about to be taught, a given contest may be more suited than another one. This work proposes a way to classify online contests according to how they can be used to support trainings and learning. This work also proposes an online platform to be used to track the performances of contestants on various online contests, in order to support their learning.

The remainder of the paper is structured as follows. The next section presents related works about classifying programming contests and about using online programming contests to support teaching and learning programming. The third section presents the proposed classification of programming contests and reviews the main existing online programming contests according to the proposed classification. The fourth section proposes a way to use online programming contests for programming trainings, based on an experience that has been set up in Belgium; and how it can be used to teach informatics. Finally, the last section presents an online platform, currently being developed, that allows anyone to have an online contestant profile to be used to support the use of online programming contests for trainings and learning.

2. Related Work

An approach to classify computer science contests has been proposed in (Pohl, 2006). In that work, the author proposes a classification scheme used to support the discussion about computer science contests. That work focuses on competitions for high school students. Six classification dimensions are proposed: scientific area, style, duration, grading, submission and divisions. Pohl (2006) also states that when participation and achievement in a contest are published, it increases the motivation and fun for the contestants. This latter statement motivates the creation of the online platform for contestant profiles proposed in this paper.

A succinct programming competitions overview is presented in (Forišek, 2013). The author highlights that most programming contests, at least those covered in his paper, are focused on the design of efficient algorithms to solve given problems. In his paper, Forišek (2013) presents tasks that have been used in past contests and that cover other areas of computer science than the one of writing efficient algorithms. The lesson of that paper is that the focus can be placed on other aspects than algorithms efficiency, and that there is consequently a place for learning about many other aspects of computer science through contests. This is in fact exactly the purpose of the Bebras contest, which aims at increasing computer fluency by secondary schools students (Futschek *et al.*, 2009).

Ragonis (2012) explores in her paper the large variety of questions that are used within the computer science (CS) discipline, and in particular she discusses in what ex-

tent those questions can be used for different teaching situations and processes, depending on the type of question. Questions proposed in the main programming contests can also be classified according to the types proposed by that author.

Programming problems similar to those used in competitions can be used to promote informatics as discussed in (Voigt *et al*., 2010). That paper presents how competition-style programming problems can be combined with *CSUnplugged* activities (Bell *et al.*, 2009) and how they can connect together programming and computer science concepts. In their work, the authors conducted tests, which show that adding programming to the *CSUnplugged* activities deepens the understanding of the concept behind the activities among students. That result strengthens the intuition that programming helps to get a more thorough understanding of computer science concepts.

Using programming contests to increase programming skills among secondary school students has already been explored in a project presented in (Nowicki *et al.*, 2013). That paper presents programming courses that are taught using OLAT distance learning tools (OLAT, 2014). Students are monitored online through weekly programming contests. The conclusion of their work, based on a test of the project in which over 900 participants attended the activities, is that it increased programming and algorithmic skills of the participating pupils. Also, as described in (Audrito *et al.*, 2012), contests (and not only online ones) do have an effect on informatics education as it creates a movement starting in the schools.

Programming contests also help to make learning programming fun as discussed in (Garcia-Mateos *et al.*, 2009). The authors describe an e-learning experience where they used programming contests as an activity to replace the final exam for a second-year programming course for computing majors at university. The contests have been set-up with the Mooshak (Leal *et al.*, 2003) automatic judging system. The results presented in that paper showed that the approach increased self-assessment skills among the students.

Using online platform to support the teaching of programming is also becoming widespread nowadays as testified by a bunch of recent research (Combéfis *et al.*, 2012; Helminen *et al.*, 2009). Those different works have in common that they use techniques to automatically assess the code produced by the learners. Moreover, as highlighted by Combéfis *et al.*, not only automatic assessment is important, but also good quality feedbacks, such as those supported by the Pythia platform (Combéfis *et al.*, 2012).

The presented related work can be summarised with three concepts: classifying contests, promoting computer science through contests and improve learning of programming with contests. Those concepts are precisely focuses of the work presented in this paper. Another key element to remember from this related work is that using contests in combination with various activities may increase programming, algorithmic and self-assessment skills and can be fun. Finally, as discussed in (Hassinen *et al.*, 2006), the best way to learn programming is through programming and extensive practice. Programming being the language of technology (Cohen *et al.*, 2007) and in particular of informatics, and that latter one being everywhere nowadays (Verhoeff, 2013), it is even more important to work on ways to improve the teaching and learning of programming, and promoting it amongst people.

3. Classifying Online Programming Contests

This section proposes classification criteria for online programming contests. The criteria can be used to compare the contests. They can also be used to make it easier to choose whether a given contest is suited for programming trainings or teaching. The proposed criteria are inspired from those of Pohl (2006), but were extended and, additionally, a distinction has been made between criteria related to the contest in general and those related to the tasks proposed in the contest.

3.1. *Classification Criteria*

The first set of criteria is related to general information about the contest. The first criterion (**I1**) is about whether the contest is restricted to single contestants or they have to participate as a team (Table 1). The second criterion (**I2**) is about the conditions that contestants must satisfy related to their age, gender or any other constraints related to their study year. The third criterion (**I3**) is about the programming languages that are accepted. The fourth criterion (**I4**) is about the duration of the contest, or the timespan during which they are allowed to work on the tasks and submit their solutions. The fifth criterion (**I5**) is about the frequency of the contest. Note that some contests are always open and start as soon as you decided to start it. They are referred to as open contests. Finally, the last criterion (**I6**) is about how the scores of the contestants are computed for their submissions, in order to establish the ranking of all the contestants.

The second set of criteria is related to the tasks the contestants have to solve during the contest (Table 2). Of course, one given contest may mix several kinds of tasks, even if it is not generally the case, at least for the contests covered in this paper.

The first criterion (**T1**) is about the type of submission that the contestant has to provide, that is, a source code, an executable or just a text file with the output produced by his/her program. The second criterion (**T2**) is about the type of task, that is, whether the contestant has to write a function whose specification is given, to model and solve a problem, to write an artificial intelligence… The third criterion (**T3**) is about any limitation on the number of trials allowed, and also about any limitations related to the execution time or the maximal allowed memory. The fourth criterion (**T4**) is about the

Table 1
Information criteria

I1	Team	Single contestant or teams
I2	Age and gender	Ages range required to participate and accepted genders
I3	Language	Accepted programming languages
I4	Duration	Timespan during which contestants can submit solutions
I5	Frequency	Frequency with which the contest is organised or open
I6	Scoring	How the score of the contestant is computed

Table 2
Task criteria

T1	Submission	Code source, executable program, output data
T2	Type	Writing a function given specification, solving a problem, writing an artificial intelligence
T3	Limitation	The number of trials that are allowed, time and memory
T4	Feedback	The feedback produced for a submission
T5	Level	Any difficulty level or partition of the tasks

feedback that is produced when the contestant makes a submission. It can go from no feedback at all, to an indication about any compile errors, execution errors, test failed errors… Finally, the last criterion (**T5**) is about whether the tasks of the contest are partitioned, according to difficulty levels, for example.

Compared to the classification proposed in (Pohl, 2006), some of the proposed dimensions have been fixed to a single value for the contests that are relevant to the purpose of this paper:

- The scientific area of the contests is limited to those with a focus on algorithmic.
- Only contests with an automatic grading are considered.
- And finally, the submissions for the considered contests are limited to software (executable or source code) or answer value.

3.2. *Review of Existing Online Contests*

This section reviews several online programming contests, and positions them according to the classification criteria proposed in the previous section. The review is not meant to be comprehensive but covers the main online contests.

Internet Problem Solving Contest (IPSC) is a contest for teams that can contain up to three people. Contestants have to solve problems, by finding the outputs that correspond to given inputs to the problems. They can write programs to solve the problems but it is not always necessary to do so. The contest has been organised yearly since 1999 and is opened to everyone, but with a special category for teams out of secondary schools.

ACM International Collegiate Programming Contest (ACM-ICPC) is an international contest, made of several stages: local, regional and then international. The contest is opened to university students that have to participate as teams. The contestants receive problems that they have to solve, providing a program written in C, C++ or Java. Moreover, the problems of ACM-ICPC are available after the contest on the *UVa Online Judge platform*, which makes it possible to try to solve them at any time, in the same conditions and with the same grader than the one used during the ACM-ICPC contest.

IEEE also proposes a contest, namely the *IEEEXtreme Programming Competition*, which lasts 24 hours and is dedicated to teams of students. All the teams receive a set of programming problems and as for ACM-ICPC, they have to solve the greatest number of problems.

Google Code Jam is a contest put in place by Google to identify potential persons to recruit. The contest consists of a set of algorithmic problems that must be solved within a time limitation. Any programming language is accepted since the contestants just have to provide the outputs corresponding to generated inputs for each problem. The contest has several rounds, all taking place online all over the world, except the worldwide final, which is hosted in one unique location for all finalists.

A lot of countries do have online programming contests to make the selection for their national team to be sent to the *International Olympiad in Informatics* (IOI). Most of those contests are opened to anyone in the world, not to compete but to participate. Just to mention some of them: *USA Computing Olympiad* (USACO), *France-IOI*, *Croatian Open Competition in Informatics* (COCI), *French-Australian Regional Informatics Olympiad* (FARIO)… All those contests are following the same philosophy as the IOI.

ProjectEuler is a collection of challenging mathematical and computer programming problems that cannot be solved only with mathematical insight. It is not exactly a contest as the previous ones. Rather, people can connect on the website at any time and solve the different problems to increase their position in the ranking.

Finally, *CodeChef* is an online contests hosting platform. The platform proposes contests regularly (short ones and also long term contests) that are open to everyone. A very large number of programming languages are accepted. The contestants have to submit their source code that is automatically graded by the platform.

3.3. *Classification of Contests*

Table 3 summarises the review of the selected contests presented in the previous section, according to the classification criteria proposed in Section 2. Information concerning the IOI is not all true for all the IOIs that already took place. The specific information shown in the table comes from the rules of IOI 2013. In particular, criteria **I6**, **T1**, **T2** and **T3** are not true for all the IOIs.

Looking at the table testifies that there are several ways to classify online contests depending on what is the focus and goal of the comparison. Here are examples of possible classifications:

- One could be interested in contests that can help to improve teamwork skills. In that case, the only criterion to look at is **I1**.
- One can be interested in contests to support learning programming, and not being an expert in algorithm designs. Looking at **T3** provides clue about the limitations that may mean that the focus in on performance and complexity (penalties for wrong submissions and time taken) and **I6** provides the conditions to win, which is whether the focus is on correctness or efficiency. Criterion **T5** can also bring information, in particular if the contest proposes subproblems with increasing complexity/size.
- Another possible way to classify contests is with respect to the feedback that is produced for each submitted solution. The **T4** criterion contains that precise infor-

mation. It is important to have good feedback if the goal is to allow the contestants to get a better learning while diminishing the intervention of trainers.

- If contests are to be used to build a training, as explained in the next Section, it is important to have some regularity (**I5**) and to have enough accepted programming languages (**I3**) if the training is to be general, or to have the taught programming language accepted if the training is to be specific.

Another review of programming contests is proposed in (Forišek, 2013). The considered contests are not restricted to online ones. Four categories to classify the considered contests are proposed by the author:

- ACM-ICPC.
- IOI.
- Company-branded contests such as Google CodeJam.
- And finally large portals hosting contests regularly such as CodeChef.

That classification, even if very simple, summarises well the actual situation. The two first contests are old worldwide and well-established ones. They are mainly focused on efficient algorithms designs and have a limited number of accepted programming languages. They both consist in a set of problems with as goal to solve the maximal number of them in a given amount of time. Both contests require the contestant to submit a program that will be executed for automatic grading. Finally, in both cases only a very limited feedback is proposed to the contestants for a given submission.

The third category, namely company-branded contests, corresponds in fact to a tool used by those companies to help them in their recruit process. In addition to Google CodeJam, contests in this category include Facebook Hackaton, for example.

Finally, the last category corresponds to platforms that host contests. Those platforms are themselves managing and proposing contests but also allow anyone to create his/her own contest. In addition to CodeChef, contests in this category also include TopCoder, for example Table 3.

Table 3

Review of selected contests according to the proposed classification criteria

Internet Problem Solving Contest (IPSC)	**I1** Teams of up to three people	**T1** Output data
	I2 Open to everybody, separate ranklists for individuals and teams in the secondary school division	**T2** 10 to 20 problems to solve, provided input and output specifications, and examples
	I3 N/A	**T3** 10 submissions at most for each subproblem (unless declared otherwise)
	I4 One block of five hours	**T4** Correct or wrong
	I5 Once every year, since 1999	**T5** Easy and hard input data
	I6 Winner is the team with the most points received; criteria taken into account are time, number of wrong submissions and difficulty level	

ACM International Collegiate Programming Contest (ACM-ICPC)	**I1**	Team of three people	**T1**	Source code
	I2	Basically students enrolled in a degree program at the sponsoring institution at least a half-time load and having left secondary school for less than 5 years	**T2**	At least 6 for regional and at least 8 for world final problems to solve, provided input and output specifications, and examples
	I3	Java, C, C++	**T3**	20 penalty minutes per wrong submission; execution time limit for problems
	I4	One block of about five hours	**T4**	Accepted or Rejected (run-time error, time-limit exceeded or wrong answer)
	I5	Once every year, with several stages: regional contests then world final, since 1978 (the first edition being on 1974)	**T5**	-
	I6	Winner is the team with the most problems solved; criteria taken into account are earliest time of submittal of correct submission and number of wrong submissions		
IEEEXtreme Programming Competition	**I1**	Team of up to three people (max 2 graduate students)	**T1**	Program
	I2	IEEE members (student or graduate student)	**T2**	A set of programming problems
	I3	C, C++, C#, Java, Python, Ruby, Perl, PHP	**T3**	No limit
	I4	One block of 24 hours	**T4**	?
	I5	Once every year (edition 7.0 in 2013)	**T5**	Easy, Medium and Hard problems
	I6	Winner is the team with the most points; criteria taken into account are: difficulty evaluated as the number of other teams who succeeded the problem		
Google Code Jam	**I1**	Individuals	**T1**	Output data + source code
	I2	Open to everybody	**T2**	A set of problems, provided input and output specifications, and examples
	I3	N/A	**T3**	4 min for small input data and 8 min for large ones
	I4	Four online rounds and one on-site world final	**T4**	Message for malformed or over-sized submission; Correct/Failed for small input data and no feedback for large input data
	I5	Once every year (since 2003)	**T5**	Small/Large input data
	I6	Winner is the contestant with the most points; criteria taken into account are the number of correct submissions and time		

International Olympiad in Informatics (IOI)	**I1**	Individuals	**T1**	Source code
	I2	Secondary school students enrolled during the period September to December in the year before IOI'*n* and is not older than 20 on the 1st July of the year of IOI'*n*	**T2**	Two times three tasks, provided input and output specifications, and examples
	I3	C, C++, Pascal	**T3**	One submission per minute and at most 100 submissions for a task; execution time and memory limit for the tasks
	I4	Two blocks of five hours	**T4**	Solved or Not solved (Incorrect solution, Run-time error/Out of Memory or Time limit exceeded)
	I5	Once every year, since 1989	**T5**	-
	I6	Winner is the contestant with the most points; criteria taken into account are the number of subproblems solved		
ProjectEuler	**I1**	Individuals	**T1**	A number
	I2	Open to everybody	**T2**	468 problems to solve, provided a description
	I3	N/A	**T3**	N/A
	I4	N/A	**T4**	Correct or wrong
	I5	Always opened	**T5**	-
	I6	Ranking based on the number of problems solved		
CodeChef	**I1**	Individuals	**T1**	Source code
	I2	Open to everybody	**T2**	Generally between 4 to 12 problems
	I3	Many languages among which C, C++, C#, Erlang, Haskell, Java, Pascal, Perl, PHP, Python, Ruby, Scala	**T3**	Execution time limit for problems
	I4	Generally between 2 and 3 hours, and a week for long challenge	**T4**	Accepted or Rejected (Time limit exceeded, wrong answer, runtime error or compilation error)
	I5	At least one contest a month (since 2009)	**T5**	-
	I6	Ranking based on the number of problems solved		

4. Using Existing Online Contests for Trainings and Teaching

As introduced above, the main goal of the programming contests mentioned so far is to allow contestants to compete against each other, alone or with a team, in order to get the best score. This section proposes two other ways to use online programming contests, to support programming trainings or teaching of informatics. The section also presents an

online platform, currently being developed, where people can create contestant profiles to be shared and compared with others.

4.1. *Programming Trainings*

The *Belgian Olympiad of Informatics* (be-OI) is already using existing online contests during the team selection process for the International Olympiad in Informatics (IOI). The be-OI is composed of several stages: several local semi-finals, one national final and an IOI selection process (Combéfis *et al.*, 2011). More precisely, for the IOI selection process, a certain number of contestants who got the best scores for the final are invited to join a pool of contestants, from which the four Belgian representatives for the IOI are selected.

Pupils from the pool were asked to participate to various selected online programming contests, and the scores they realised were tracked and put on an online wiki so that the pupils were able to compare themselves. The scores they achieved on those contests were taken into account in the IOI team selection process. Achieving good scores for a pupil of course increases his/her chance to be selected. Since it is not possible to check whether pupils were helped or not by external people, the selection process includes an additional small contest for which all the pupils from the pool are physically in the same room at a same moment.

The experience that was set up with the be-OI was positive for the coaches as well as for the pupils from the pool. Generally speaking, several advantages can be identified for building programming trainings based on existing online contests:

1. It reduces the human resources needed to coordinate the training.
2. It suppresses the need to create programming exercises and tasks.
3. It allows pupils to code a lot and therefore to improve in some way their coding efficiency with practical exercises that they can do at home.
4. It allows pupils to compare themselves with other worldwide contestants.

Using such a programming training for the IOI team selection process is therefore relevant when the available human resources are rather low. It saves them time to take care of other interesting aspects of the training. For example, the coaches can use their time to review with the pupils some of the exercises proposed in the selected online contests, to teach them new concepts or techniques to maybe better solve those exercises. Such a consideration is the subject of the next section.

There are also disadvantages of using existing online contests for programming trainings. The main ones that can be observed are:

1. It is very difficult to motivate pupils to spare time to participate in all those contests for which the hours where it is possible to participate are not always convenient.
2. It is honour based, meaning that there is no control on whether it is actually the pupil who participated to the contest, which makes it impossible to use for a selection process.

As discussed in section 4.3, the proposed online platform for contestant profiles aims at overcoming the first disadvantage, by increasing their motivation such as described by Pohl (2006), by publishing their scores publicly or in a group of selected pupils.

Not all contests are best suited to be used for programming trainings. Since the goal is essentially for individual trainings, only contests where individuals can participate should be considered (**I1**). If the contest is to be used to rank people, it is important to carefully look at (**T6**) in order to vary the different kinds of evaluations, just for the same reason for varying the types of questions exposed in (Ragonis, 2012). The accepted languages (**I3**) are also important if the goal of the training is to improve the skills of a specific language or to prepare the learners to a specific contest.

4.2. *Teaching Informatics*

As detailed in the previous section, using online contests for programming trainings does not provide any guarantee about the skills the contestant will actually learn from the training. That latter fact is reinforced when the feedback provided to the contestants for their submissions is of poor quality. However, this does not mean that it is impossible to teach informatics with the support of online contests for some activities, as it has been highlighted in the related work section.

Limiting the activity of the pupils to just participating in the contest is not enough to get a good learning. Additional activities supervised by trainers must be proposed in addition to the contest. As explained in (Combéfis *et al.*, 2012), when it is to learn programming, the feedback that is given to learners is very important to support their learning. The additional activities that have to be proposed to teach informatics with online contests are thus centred on feedback. In case of online programming contests, the additional proposed activities or material to provide are:

1. Solutions to the problems, the more detailed and annotated, the best.
2. Feedback about the solution of the learner, that is, information about why it is not correct or which elements can be improved.

Most of the time, online contests do not take into account any qualitative consideration about the code submitted by contestants. As highlighted in (Forišek, 2013), most traditional programming contests only focuses on the design of efficient algorithms. As it can be observed in the review of the main online programming contests, the feedback that is provided for each submission is either non-existent or is very limited.

Using online programming contest to do "real-time" teaching of informatics is certainly not appropriate due to the lack of feedback. However, and as the project presented in (Nowicki *et al.*, 2013) testifies for example, it is still possible to combine programming contests with a given educational device in order to improve the quality of learning. To do that, it is important to provide feedback to the learners. The platform presented in section 4.3 can be used to attach feedbacks to problems of various online contests, so that contestants can learn from the problems after having tried to solve them.

4.3. *An Online Platform to Share Contestant Profiles*

This section describes the *My Contestant Profile* (MCP) online platform that can be used to support programming trainings and teaching of informatics by using online programming contests*.

The purpose of the platform is to maintain a list of online programming contests, and to allow its users to have a contestant profile on it. As soon as a user participated to an online contest (or a round of a contest), he can go on his profile and add the score he performed. The main benefit of doing so is the possibility for the user to compare his score with his friends, people from the same city/region/country or any other relevant group of people that could be defined. That first feature can be used to monitor programming trainings. A specific group of people can be set up on the platform so that the coach can monitor the performance of the users belonging to the group. It could be used for the be-OI contest, for example, defining a group containing the contestants from the pool.

The second important feature of the platform is the discussion forums used to discuss about a contest that took place, or more precisely on the problems of a given contest. It is possible through the platform to attach feedback information and detailed solutions for each problem. Letting users discuss amongst themselves enriches their understanding.

The aim of the MCP online platform is to provide a tool to support the two previously uses of online contests, namely programming trainings and teaching of informatics. The first feature described is used to manage a pool of trainees and to improve their motivation. It allows trainers to monitor the contestants, and to use their performance as one indicator for a selection process. The second feature described is used to support teaching of informatics. It proposes a place where additional materials related to the tasks and problems of contests can be posted and especially discussed between trainers/teachers and contestants/learners, but also between the users of the platform.

5. Conclusion and Perspectives

This paper takes the opportunity that there are more and more freely accessible online programming contests to discuss the possibility to use them to build programming trainings and to teach informatics. The paper first proposes a short review of the main online programming contests and describes them according to a proposed classification. The classification is built around two sets of criteria. The first set contains criteria on general information about the contest such as the conditions of admission or the accepted programming languages. The second set contains criteria about the tasks/problems used in the contest such as the type of submission or time and memory limitations.

Whereas using online programming contests for programming trainings has already been experimented with in Belgium, using them in the context of teaching informatics has only been tested by some researchers as described in the related work section. This paper highlights that online programming contests can be used to build programming

* The online platform is available on: `http://mcp.csited.be`

trainings and to support teaching informatics. In order to support those two latter uses of contests, the paper presents an online platform to host contestant's profiles.

Future work includes deploying and populating the MCP platform with contests and problems, and to produce feedback information about those problems. Based on that, experiences to build trainings or to teach informatics have to be set up and monitored to assess whether they improved programming skills. Another way of investigation is to measure the increase of motivation among users of the MCP platform.

Perspectives include developing a way to measure the impact of the platform on the motivation to participate in more contests and learn programming. Experiments have to be done on several groups of people: contestants that are to be trained for a specific contest (IOI, for example), pupils at school that have a programming class and are learning to program, and finally people at large studying programming and willing to improve their skills and share their thoughts about tasks/problems found in online contests.

References

Audrito, G., Demo, G., Giovannetti, E. (2012). *Olympiads in Informatics*, 6, 3–20.

Bell, T., Alexander, J., Freeman, I., Grimley, M. (2009). Computer science unplugged: school students doing real computing without computers. *Journal of Applied Computing and Information*, 13(1), 20–29.

Cohen, A., Haberman, B. (2007). Computer science: a language of technology. *SIGCSE Bulletin*, 4(39), 3–14.

Combéfis, S., Leroy D. (2011). Belgian olympiads in informatics: the story of launching a national contest. *Olympiads in Informatics*, 5, 131–139.

Combéfis, S., le Clément de Saint-Marcq, V. (2012). Teaching programming and algorithm design with pythia, a web-based learning platform. *Olympiads in Informatics*, 6, 31–43.

Forišek, M. (2013). Pushing the boundary of programming contests. *Olympiads in Informatics*, 7, 23–35.

Futschek, G., Dagiene, V. (2009). A contest on informatics and computer fluency attracts school students to learn basic technology concepts. In: *Proceedings of the 9th World Conference on Computers in Education (WCCE 2009)*.

Garcia-Mateos, G., Fernandez-Aleman, J.L. (2009). Make learning fun with programming contests. In: *Transactions on Edutainment II*. Lecture Notes in Computer Science 5660, 246–257.

Hassinen, M., Mäyrä, H. (2006). Learning programming by programming: a case study. In: *Proceedings of the 6th Baltic Sea Conference on Computing Education Research (Koli Calling 2006)*. 117–119.

Helminen, J., Malmi, L., Korhonen, A. (2009). Quick introduction to programming with an integrated code editor, automatic assessment and visual debugging tool – work in progress. In: *Proceedings of the 9th International Conference on Computing Education Research (Koli Calling 2009)*. 59–62.

Leal, J. P., Silva, F. (2003). Mooshak: a web-based multi-site programming contest system. *Software: Practice and Experience*, 33(6), 567–581.

Nowicki, M., Matuszak, M., Kwiatkowska, A., Sysło, M., Bała, P. (2013). Teaching secondary school students programming using distance learning: a case study. In: *Proceedings of the 10th World Conference on Computers in Education (WCCE 2013)*.

OLAT Team. (2014). *OLAT 7.8 – User Manual (1.2014 v7.8)*. `http://www.olat.org`

Pohl, W. (2006). Computer science contests for secondary school students: approaches to classification. *Informatics in Education*, 5(1), 125–132.

Ragonis, N. (2012). Type of questions – the case of computer science. *Olympiads in Informatics*, 6, 115–132.

Verhoeff, T. (2013). Informatics everywhere: information and computation in society, science, and technology. *Olympiads in Informatics*, 7, 140–152.

Voigt, J., Bell, T., Aspvall, B. (2010). Competition-style programming problems for computer science unplugged activities. In: E. Verdu, R. Lorenzo, M. Revilla, L. Regueras (Eds.), *A New Learning Paradigm: Competition Supported by Technology*. Boecillo: CEDETEL, 207–234.

S. Combéfis obtained his PhD in engineering in November 2013 from the Université catholique de Louvain in Belgium. He is also finishing an advanced master in pedagogy in higher education. He founded the Belgian Olympiad in Informatics (be-OI) with Damien Leroy in 2010. In 2012, he introduced the Bebras contest in Belgium and at the same time he founded the CSITEd non-profit organisation that aims at promoting computer science in secondary schools. He is interested in computer science education, and in particular on how to increase computer fluency among people at large, and also on how to build and design online programming courses that support learning as best as possible.

J. Wautelet is studying computer science at Université catholique de Louvain. He is now a first year bachelor student. He is actively involved in the Bebras Belgium project. He is also currently member of the CSITEd non-profit organisation where he contributes on several projects including Pythia. Being freshly graduated from the secondary school, he brings a fresh view about how computer science is perceived there.